Understanding and Verifying Deep Neural Networks

Divya Gopinath

Research Scientist, RSE group NASA AMES, KBR Inc.

Outline

- Introduction
- Background
- Our Approach
- Case Studies
- Future work
- Conclusion

SafeDNN: Safety of Deep Neural Networks

https://ti.arc.nasa.gov/tech/rse/research/safednn/

- NASA project that explores techniques and tools to ensure that systems that use Deep Neural Networks (DNN) are safe, robust and interpretable

- Project Members: Corina Pasareanu, Divya Gopinath
    - Many students and collaborators

- This talk focuses on $Prophecy$[1], for formal analysis of Deep Neural Networks , specifically describing its application in understanding and verifying networks used in autonomous systems

[1]: *Divya Gopinath, Hayes Converse, Corina S. Pasareanu, Ankur Taly: Property Inference for Deep Neural Networks. ASE 2019*

Outline

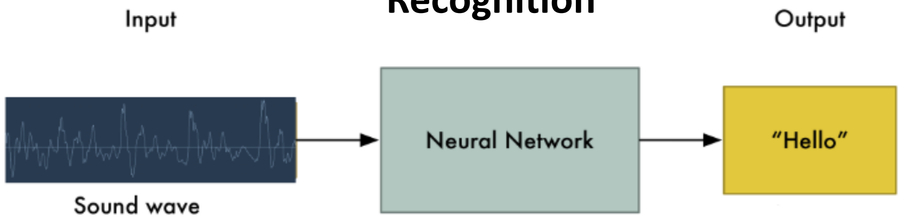
- Introduction
- Background
- Our Approach
- Case Studies
- Future work
- Conclusion

Deep Neural Networks

- Deep Neural Networks (DNNs) have widespread usage, even in safety-critical applications such as autonomous driving

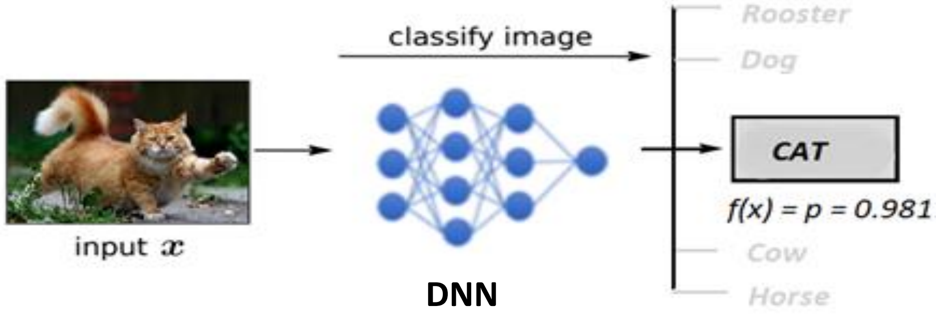
Sentiment Analysis

Speech Recognition

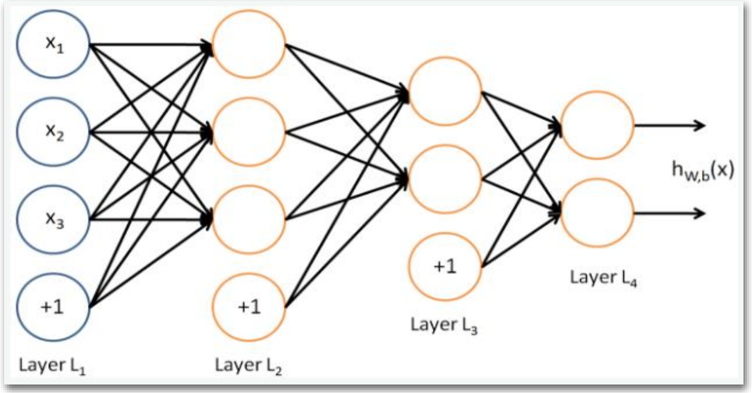


Autonomous Driving

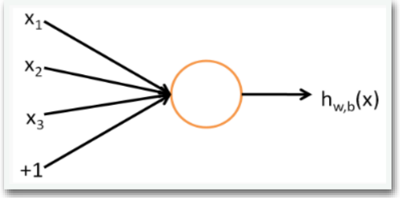
Image Classification



DNN



neuron



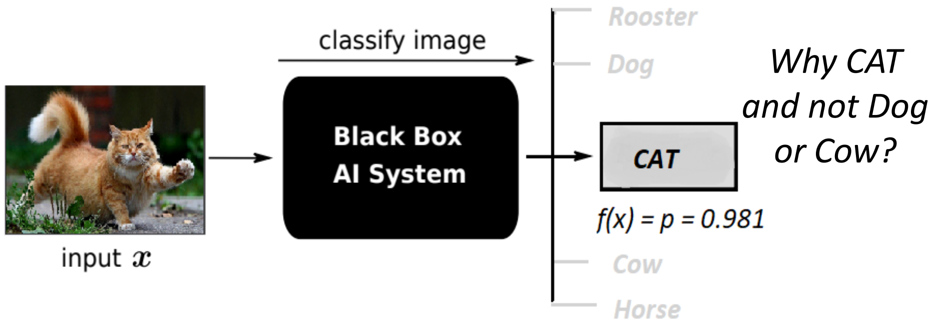
ReLU activation function

$f(x) = max(0,x)$

$h_{W,b}(x) = f(W^T x) = f(\sum_{i=1}^{3} Wixi + b)$

Challenges

- Lack of *explainability*
  - It is not well understood why the network makes its decisions
  - Design not amenable for analysis, Logic not interpretable
  - Impacts reliability, impedes trust



- Lack of *guarantees for network behavior*
  - Often networks do not have formal input-output specifications defining functional correctness
  - Networks are large and complex inhibiting efficient analysis and verification

Existing work (Explainable AI)

- Work done mostly in the fields of computer vision and NLP
- Explaining behavior of pre-trained models (Model-Specific)
    - Saliency Maps, Gradient descent, DeepLIFT, Integrated Gradients identify portions of the image that impact network prediction
    - DeepExplain, Guided-Back propagation visualize features learnt by the network at different layers
    - Class Activation Maps (GradCAM, GradCAM++) indicate discriminative regions of an image used by a CNN to categorize them into different classes
    - *Concept Activation Vectors determine how sensitive a prediction is to a user-defined concept such as a "human" or "animal"*
    - LRP is an attribution technique applicable to images and text, Rationale is an interpretability method for text (NLP)
- Explaining any black-box model (Model Agnostic): LIME, Ancor, SHAP, PDP

Existing work on Explainable AI (Open issues)

- Not much work on generating explanations for more complex output properties and behaviors than classification
- Most techniques are typically local and generate explanations wrt a single image or a set of images
- There aren’t techniques that generate formal explanations which can be proved
- There isn’t a common or generic approach that is applicable to different types of networks (classification, regression, recurrent networks so on)

Existing work (Verification)

- Number of approaches have been developed to verify if a given DNN model satisfies a property

$$\forall x \quad Pre(x) \Rightarrow Post(F(x))$$

  - For perception networks, the property is mainly Adversarial Robustness
  - For some networks (ACAS Xu; controller network with low-dimensional sensor inputs), pre-defined input-output specifications are available and have been verified
- Search-Based techniques: Reluplex, Marabou,Planet use SMT solvers such as Gurobi and Yikes
- Reachability-Based techniques: DeepZono, DeepPoly,AI2
- Optimization-Based techniques: MIPVerify
- Search+Optimization: Neurify , VeriNet

Existing work on Verification (Open Issues)

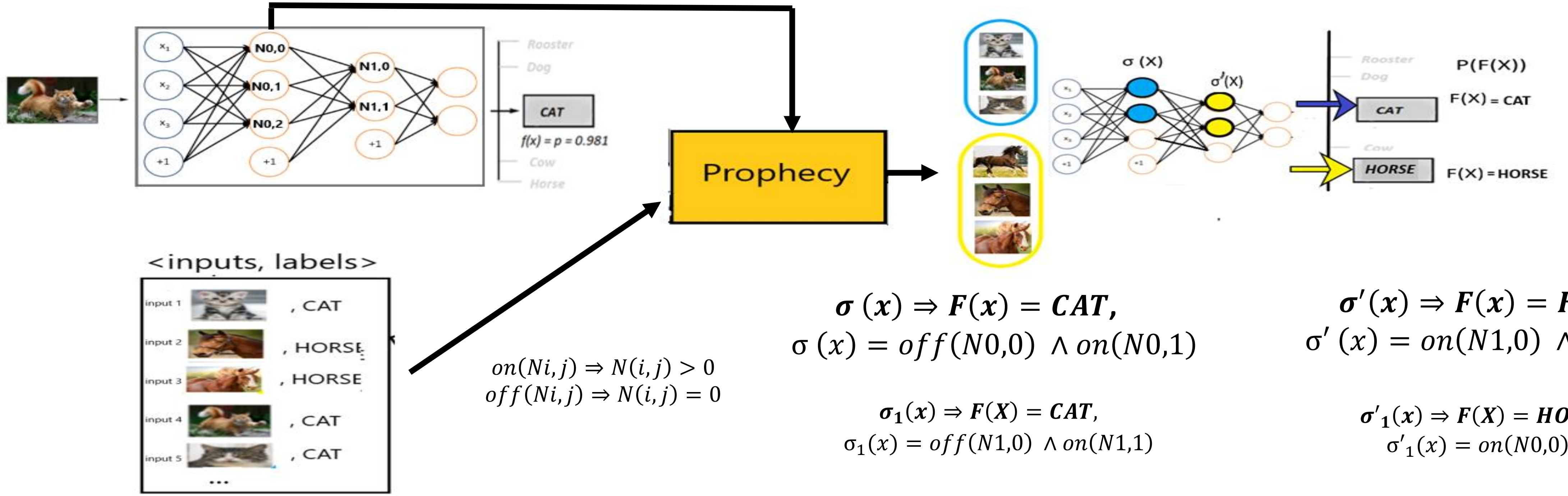
- Mainly applicable to feed-forward networks, Piece-wise linear activation functions (ReLU)
- Not scalable to large complex networks
- Guarantees of robustness in small local regions around inputs
- Need *richer*, *more expressive properties capturing the overall functional behavior of the DNN*

Outline

- Introduction
- Background
- Our Approach
- Case Studies
- Future work
- Conclusion

Our Approach

- Decompose the complex DNN model into a set of simple rules, amenable to analysis
  - Assume-guarantee type rules are inferred from a trained DNN; $\forall x\ \sigma(x) \Rightarrow P(F(x))$
  - P is a property of the network function; functional property
  - σ (X) are formal constraints on neurons at inner layers of the network *(neuron activation patterns)*
  - *Prophecy: Property Inference for Deep Neural Networks (ASE 2019)*



$on(Ni,j) \Rightarrow N(i,j) > 0$
$off(Ni,j) \Rightarrow N(i,j) = 0$

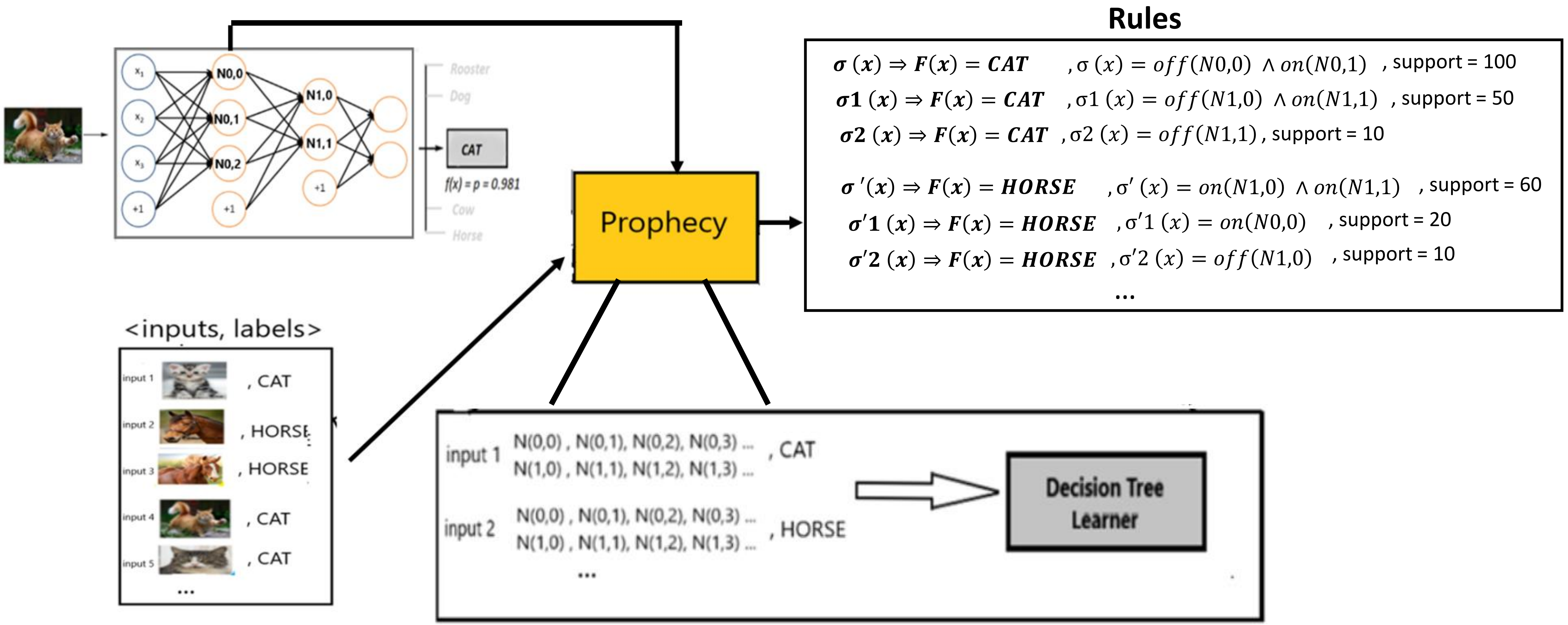
$\boldsymbol{\sigma(x) \Rightarrow F(x) = CAT,}$
$\sigma(x) = off(N0,0)\ \wedge on(N0,1)$

$\boldsymbol{\sigma'(x) \Rightarrow F(x) = HORSE,}$
$\sigma'(x) = on(N1,0)\ \wedge on(N1,1)$

$\boldsymbol{\sigma_1(x) \Rightarrow F(X) = CAT,}$
$\sigma_1(x) = off(N1,0)\ \wedge on(N1,1)$

$\boldsymbol{\sigma'_1(x) \Rightarrow F(X) = HORSE,}$
$\sigma'_1(x) = on(N0,0)$

Prophecy
Property Inference for Deep Neural Networks
[ASE 2019]

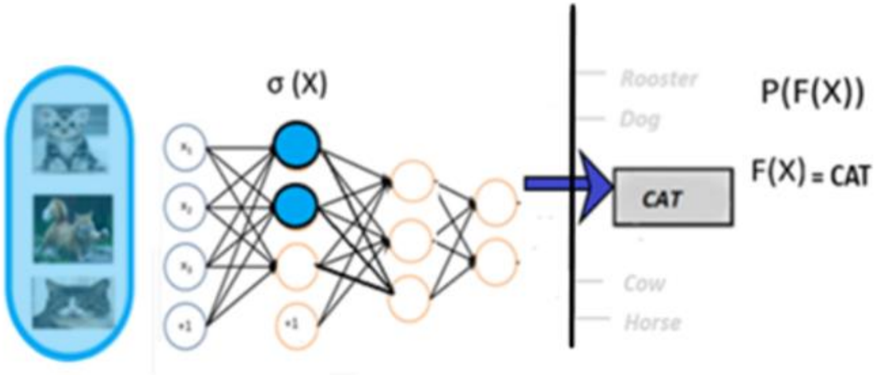


Our Approach (Benefits)

- Rules act as *"likely" specifications* , richer and more expressive properties of functional behavior
  - Faithful to network behavior
- Mathematical formulation is amenable to verification, providing *guarantees wrt functional behavior*

  $$\forall x \; \sigma(x) \Rightarrow P(F(x))$$

  - Enable more efficient (*compositional verification*) of input-output properties
- Visualization of rules enable *explainability and interpretability*
  - Obtaining *formal explanations* which can be proved
- Applicable to *any type of input* (image, text, sensory signals) and *complex output properties*



Applications

- The properties extracted using Prophecy have many applications
    - Obtaining formal guarantees of network behavior
    - Interpretability and Explainability of network behavior
    - Network Distillation
    - Proof Decomposition
    - Debugging and repair
- Case studies on perception networks, controller networks, classifier and regression models
    - Feed forward networks
    - With fully connected, convolution layers, maxpool layers
    - ReLU , eLU activation functions
- This talk will focus on our case-studies on DNNs used as *perception and controller modules in autonomous driving*

Outline

- Introduction
- Background
- Our Approach
- Case Studies
    - Regression Model for Perception
- Future work
- Conclusion

Case study on a Regression model for Perception [2]

- **TaxiNet** is a neural network designed to take a single picture of the runway as input and return the plane's position w.r.t. the center of the runway
  - Returns 2 numerical outputs; **Cross track error (y0)**: The distance of the plane from the middle line, **Heading error (y1):** The angle of the plane w.r.t. the middle line
- Input data is a sequence of images captured by the camera as the plane moves on the runway
  - A simulator (Xplane) used to generate data for training and testing

https://blog.send-anywhere.com/blog/2016/01/13/uc-3-entertainment-for-travel/boeing-aircraft-plane-on-runway-free-wallpaper-hd/

[2]: *Burak Kadron, Divya Gopinath, Corina Pasareanu, Huafeng Yu: Case Study: Analysis of Autonomous Center lineTracking Neural Networks. VSTTE21*

Problem Statement

- Desired properties of the network outputs
  - *Safety property:* In order to ensure that the plane is in the safe zone within the runway
    $|y0| < 10.0m$, $|y1| < 90°$
  - *Correctness property:* Based on data whose ideal outputs are known
    $|y0-y0ideal| < 1.5m$, $|y1-y1ideal| < 5°$

- Can we understand why the network behaves (correctly/incorrectly) in some scenarios vs. others?
  - We want to identify *input features* that impact network behavior w.r.t *correctness constraints*
  - The feature should be a *characteristic of a sequence of images*
  - Useful in debugging, generating additional testing scenarios, Runtime monitors

- Can we generate guarantees for the safe operation?
  - We want to generate *guarantees over sequence of images (or a time window)*
  - We would like to generate *new image sequences that can lead to failure*
  - Important to build trust and certify network behavior

- Can we produce sound results despite considering the network as a standalone entity without the feedback loop with the simulator?

Prophecy on TaxiNet

- TaxiNet Architectures
  - ***Boeing TaxiNet [REF]****:* CONV network with 24 layers, input is a 360x200x3 image, 5 CONV layers, 5 activation layers and 3 dense layers (100,50,10 eLU neurons respy) before the output layer with 2 outputs
  - Prophecy used to extract patterns using a labeled dataset with 13885 inputs
    - Wrt three correctness properties; $|y_0 - y_{0ideal}| \leq 1.0$, , $|y_1 - y_{1ideal}| \leq 5.0$, $|y_0 - y_{0ideal}| \leq 1.0 \wedge |y_1 - y_{1ideal}| \leq 5.0$
    - At each of the three dense layers and all of them together
    - Patterns for satisfaction (396 patterns for class 1), patterns for violation of the correctness properties (418 patterns for class 0)

  - **Tiny Taxinet [3]:** Smaller network takes in a down-sampled version of the image (128 pixels), 3 dense layers (16,8,8 ReLU neurons respy) and output layer with 2 outputs
  - Prophecy used to extract patterns using a labeled dataset with 51462 inputs
    - Wrt three safety properties; $|y_0| \leq 10.0$, $|y_0| \leq 8.0$, $|y_0| \leq 5.0$
    - At each of the three dense layers and all of them together, patterns for satisfaction and violation of the safety properties were extracted

[3]: *K. D. Julian, R. Lee, and M. J. Kochenderfer, "Validation of image based neural network controllers through adaptive stress testing," arXiv preprint arXiv:2003.02381, 2020*

Patterns for Explainability

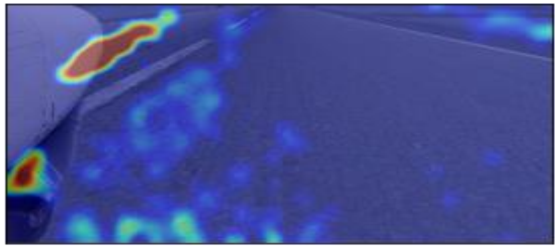
- A pattern represents **features** of the input images that impact network behavior,
  - Activation pattern from dense layer 1 for the satisfaction of the correctness property w.r.t y0 (cross-track error)

    off(N1,53) /\ off(N1,33) /\ off(N1,71) /\ off(N1,64) /\ off(N1,67) => |y0-y0ideal| <= 1.0, Support = 1792

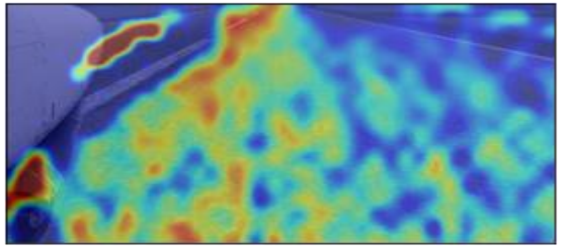
  - We visualize these features by **highlighting the input pixels** that impact the pattern
    - For an image satisfying the pattern, highlight pixels that impact the neurons in the pattern (using GradCAM++ [4])
    - Identifies portion of the image impacting network's behavior w.r.t the cross-track error output
    - Highlighting pixels that impact the output variable y0 (aka existing attribution approaches) is not as helpful

input image

pixels impacting the pattern



pixels impacting y0

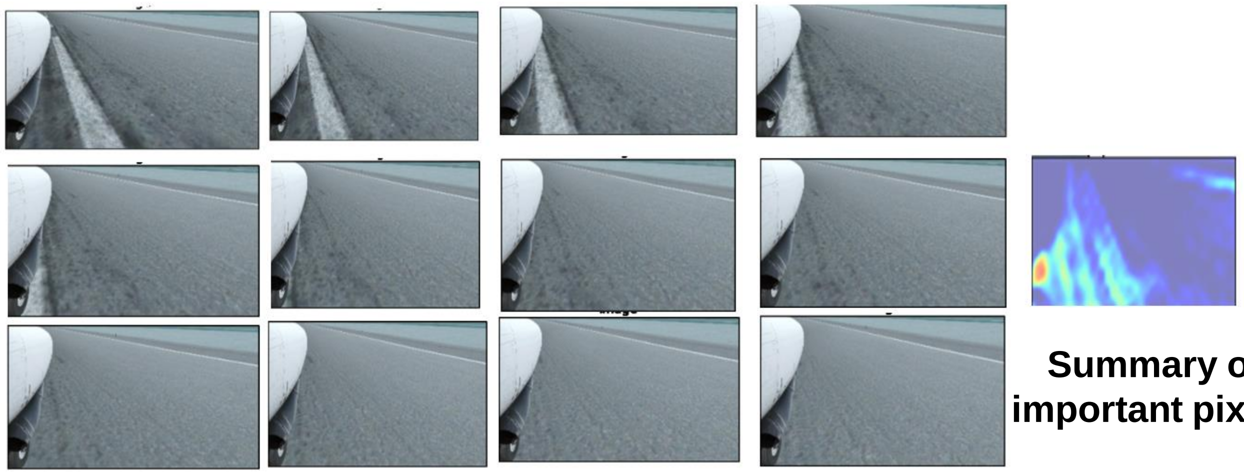


[4]: *Chattopadhay, Aditya, et al. "Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks." WACV 2018*

Explaining Correct Behavior

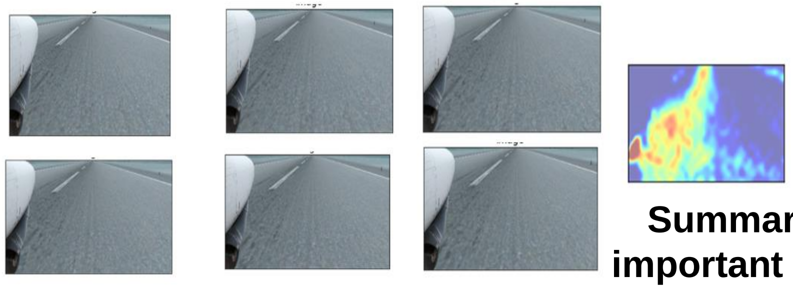
- Extracting a common characteristic (feature) over a sequence of images
  - 44 sequences (length > 5) satisfy the example pattern
  - The summary of important pixels (average GradCAM values across all images) represents the feature for the scenario that impacts the output property the most
  - The feature; distance between the center line of the runway and the airplane; is relevant for cross-track error determination enabling the network to produce the correct output for this scenario

Sequence of 12 images satisfying pattern for correct behavior



Summary of important pixels

Sequence of 6 images satisfying pattern for correct behavior



Summary of important pixels

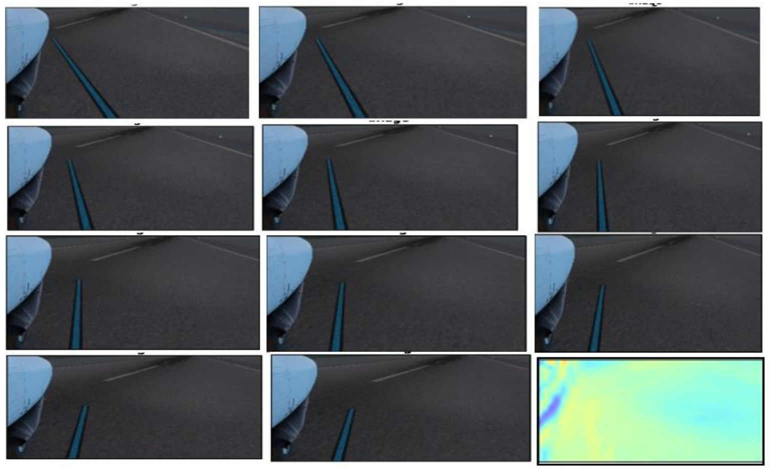
Explaining Incorrect Behavior

- Pattern from dense layer 1 for the violation of the correctness property w.r.t y0 (cross-track error)

  on(N1,53) /\ off(N1,29) /\ on(N1,20) /\ off(N1,49) /\ off(N1,15) /\ off(N1,95) /\ off(N1,25)=> |y0-y0ideal| > 1.0, Support: 403

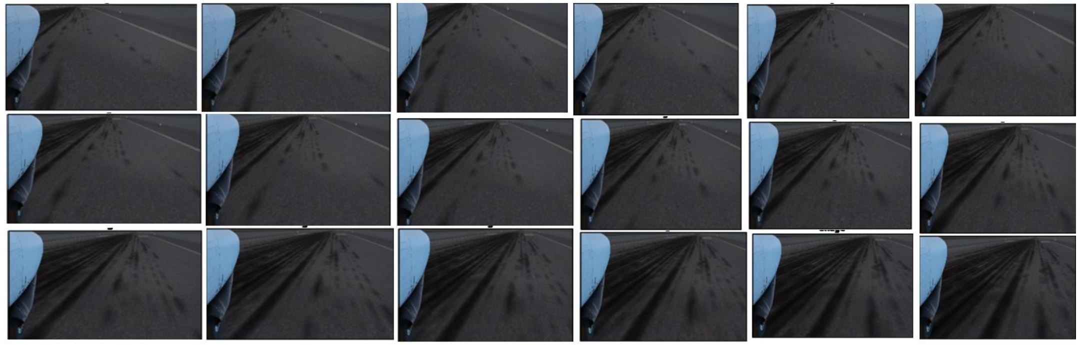
  - 18 sequences (length > 5) satisfy the example pattern
  - Scenario 1: Highlighted pixels indicate that the noise (blue line) interferes with correct determination of the cross-track error
  - Scenario 2: None of the pixels are highlighted, indicating the absence of a distinct feature that the network could use to make a correct estimation of the cross-track error

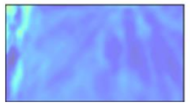
Example scenario 1



Summary

Example scenario 2





Summary

Formal guarantees of safety

- We employed the Marabou[5] solver to check if all inputs satisfying a pattern satisfy the output property
  - Formal proof of consistent behavior of the network over the input region representing the sequence of images (a time interval)
  - We were unable to use Marabou on the Boeing Model since it is unable to handle the complexity of the network, specifically the eLU activation functions
  - We were able to check the safety properties on the TinyTaxinet model using Marabou

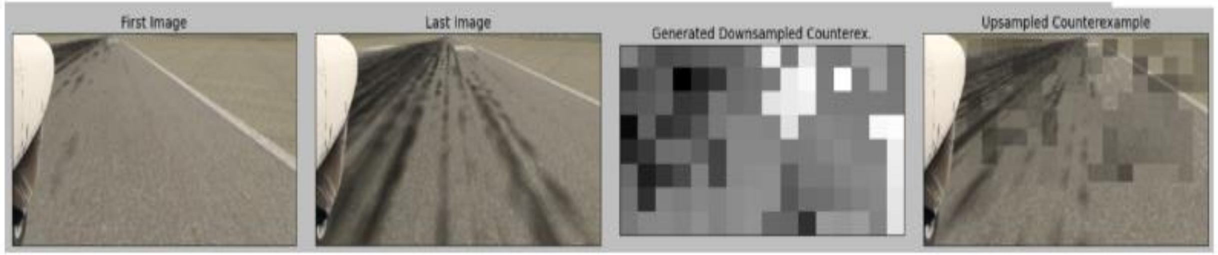
    $$\forall x \in [x_{min}, x_{max}] \wedge pattern \; => |y_0| \leq 10m$$
  - Obtained proofs for 33 sequences with at least 5 images, the longest sequence with proof had 17 images

[5] Guy Katz, Derek A. Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljic, David L. Dill, Mykel J. Kochenderfer, Clark W. Barrett: The Marabou Framework for Verification and Analysis of Deep Neural Networks. CAV (1) 2019

Counter-examples

- Generating scenarios where the plane can run out of the runway is very useful for debugging
- Counter-example to the check
  $\bigvee x \ \in [x_{min}, x_{max}] \wedge pattern \ => |y_0| \leq 10m$
- An image similar to the other valid images in the sequence but causes the network output to violate the safety property |y0| > 10m
- The inclusion of the pattern and the bounds around valid inputs in the sequence makes the counter-example more likely to occur in an actual closed-loop system

Counterexample for an image sequence of length 39 for $|y_0| \leq 10m$



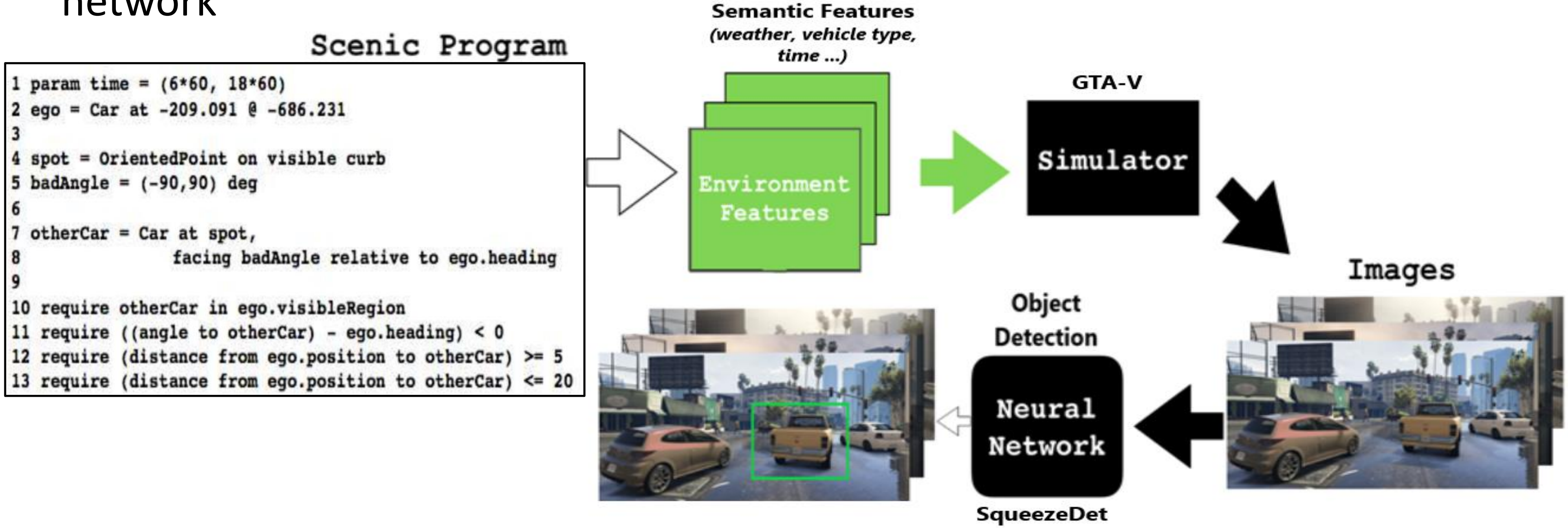
Counterexample for an image sequence of length 5 for $|y_0| \leq 10m$

Outline

- Introduction
- Background
- Our Approach
- Case Studies
  - Regression Model for Perception
  - Object Detection Network in Autonomous vehicles
- Future work
- Conclusion

Case study on an Object Detection Network [6]

- **SqueezeDet** is a convolutional neural network for object detection in autonomous cars
- **SCENIC** is a probabilistic programming language used to describe environments or scenes
  - Generates values for environment variables describing a scene using high level semantic features
- The environment variables fed to a simulator to create realistic images for the object detector network



[6] Edward Kim, Divya Gopinath, Corina S. Pasareanu, Sanjit A. Seshia: A Programmatic and Semantic Approach to Explaining and Debugging Neural Network Based Object Detectors. CVPR 2020

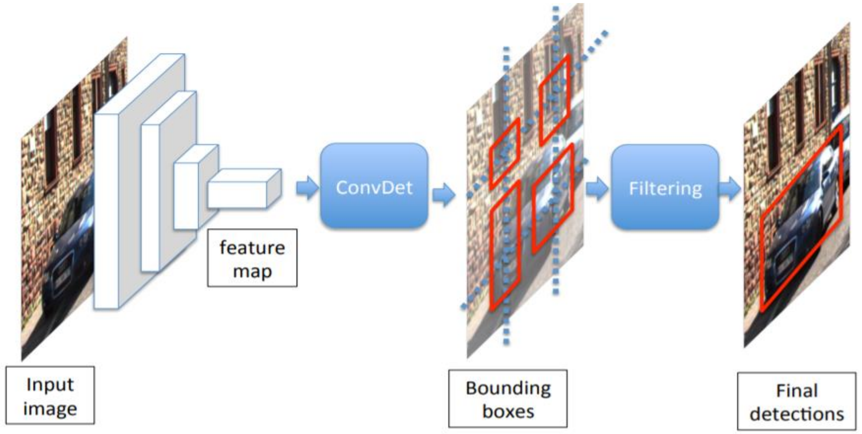
Problem Statement

- Can we generate explanations for behavior in terms of higher-level features (such as weather, vehicle type …)?
  - Most existing techniques identify important portions at a pixel level on images and require human intervention to determine what these portions correspond to in terms of a feature

- Can we generate tests that will specifically increase the correction /incorrect detection rates of the object detector, which would help with debugging?

Prophecy on SqueezeDet

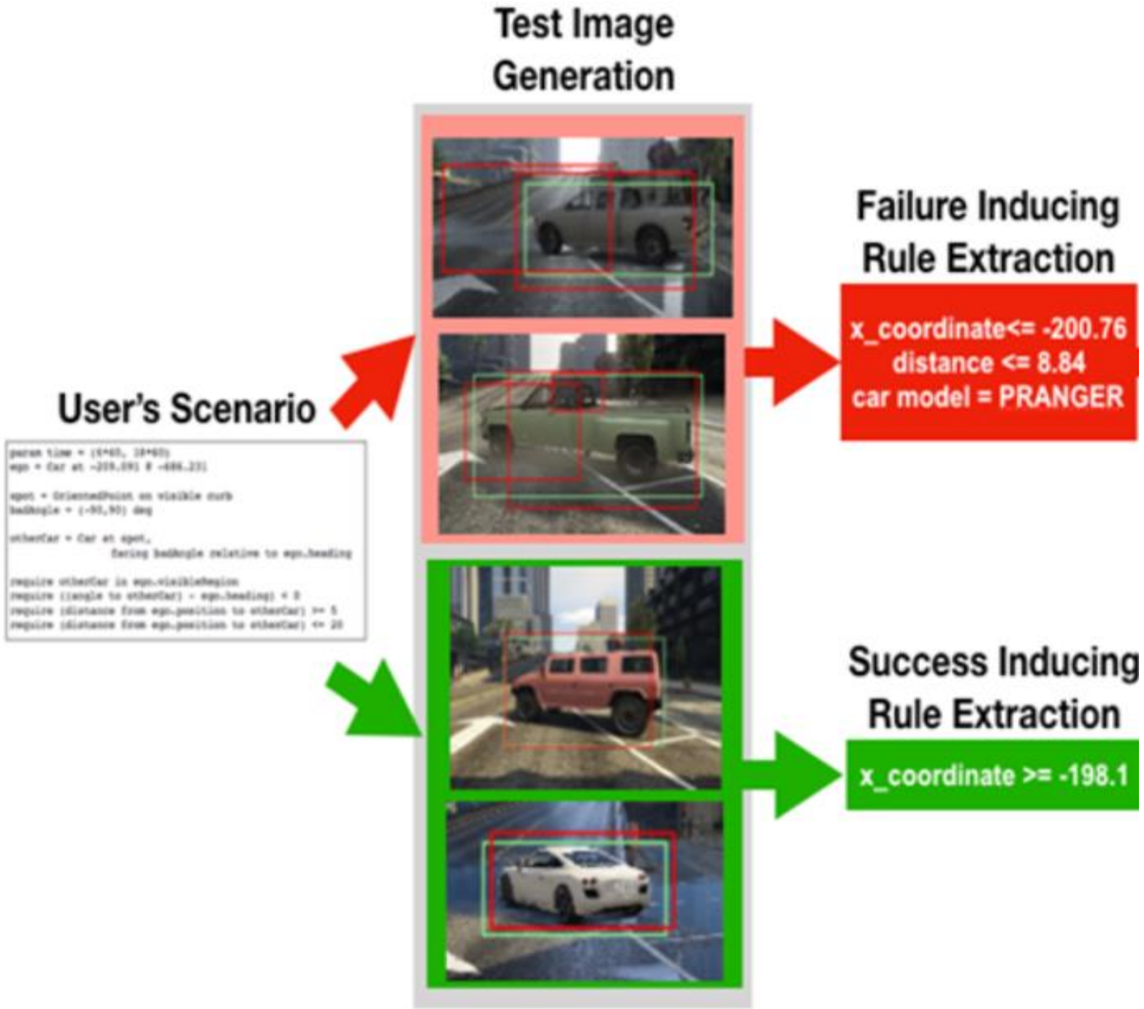
- **SqueezeDet Architecture:**



- Labelling the outputs: For each image, the network's output is labelled correct or incorrect
  - Correct Label: F1 of correct detection > 0.8, Incorrect Label: F1 of correct detection <= 0.8
  - TP: # of ground truth boxes correctly predicted (IoU>0.5), TN: # of ground truth boxes not detected, FP: # of bounding boxes falsely predicting ground truth, F1 > 0.8

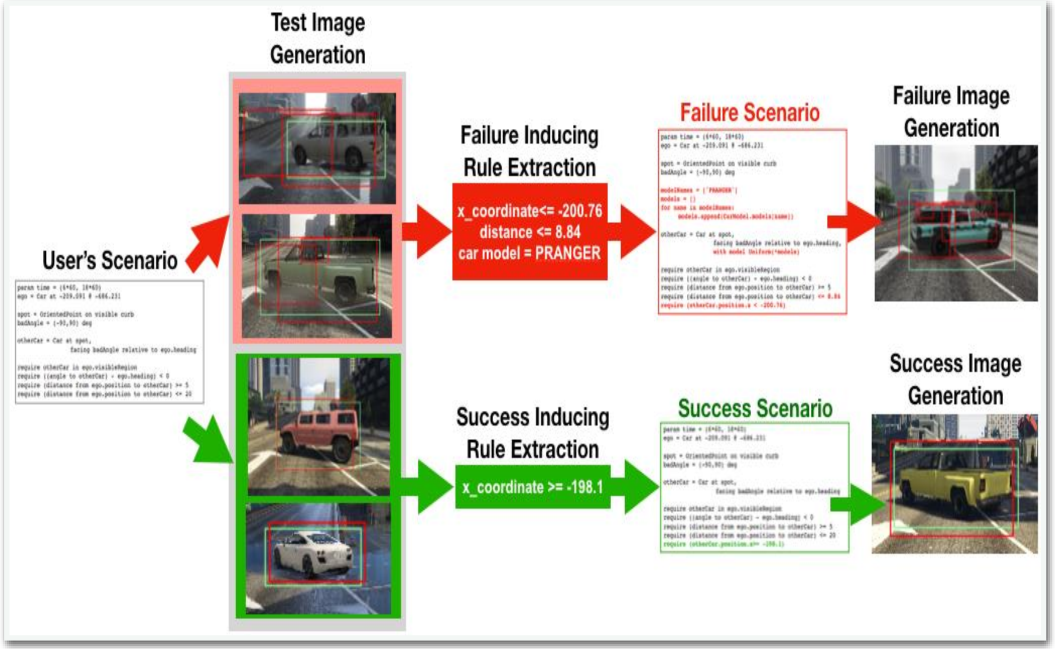
Extracting Semantic Explanations

- Given a set of labelled data, we used Prophecy to extract the neuron activations patterns from the three Maxpool layers
  - Patterns for both correct and incorrect labels were extracted
- Each input also has an associated feature vector in terms of the environment features
  - We labelled the feature vectors into 4 classes; correct-pattern, correct-nopattern, incorrect-pattern, incorrect-nopattern
  - We then used Ancor and Decision-Tree learning to extract rules in terms of the features



Generating Test Inputs for Debugging

- The failure inducing rule is used to refine the scenic program to generate more failure inducing images
- The success inducing rule is used to refine or correct the program to generate more passing tests
- Increased correct detection rate from 65.3% to 89.4% and incorrect detection rate from 34.7% to 87.2%
- These additional tests could be used to debug and/re-train the object detector network

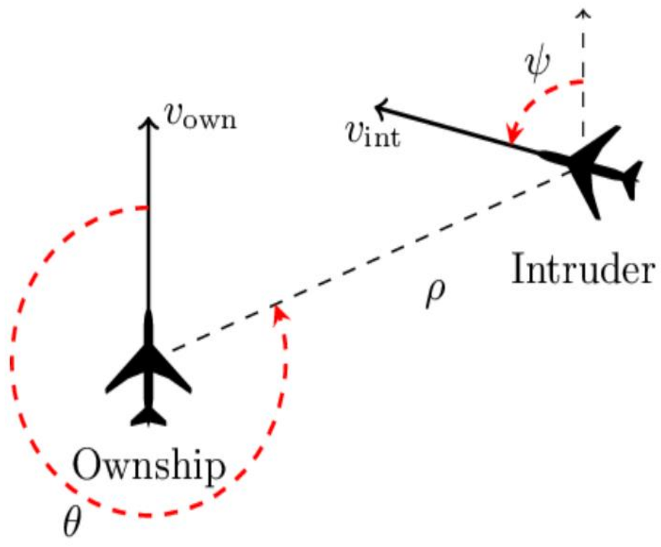


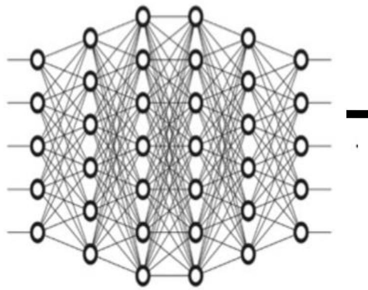
Outline

- Introduction
- Background
- Our Approach
- Case Studies
  - Regression Model for Perception
  - Object Detection Network in Autonomous vehicles
  - Controller network for Collision Avoidance
- Current/Future work
- Conclusion

Case study on a Controller network

ACAS-Xu (Airborne Collision Avoidance System-Xu)





DNN
controller

Clear of Conflict (COC)

Strong Left

Weak Left

Strong Right

Weak Right

Architecture and Properties

- 45 DNNs, Each with 5 inputs, 5 outputs, Fully connected, ReLU activations , 6 layers with a total of 300 Nodes

- System has 10 desirable properties (input-output specifications)
  - For a far away intruder, the network advises COC,
  - $36000 \leq range \leq 60760$, $0.7 \leq \theta \leq 3.14$, $-3.14 \leq \psi \leq 3.14 + 0.01$, $900 \leq vown \leq 1200$, $600 \leq vint \leq 1200$, has turning advisory COC
  - If the intruder is near and approaching from the left, the network advises “strong right”
  - $250 \leq range \leq 400$, $0.2 \leq \theta \leq 0.4$, $-3.14 \leq \psi \leq 3.14 + 0.005$, $100 \leq vown \leq 400$, $0 \leq vint \leq 400$, has turning advisory Strong Right

Problem Statement

- Existing work
  - There is a lot of work on proving adversarial robustness on this network
  - Proving the system level input-output properties such as [7] which uses the Reluplex solver to prove the input-output properties
  - Recent work explores repairing the ACASXU network with formal guarantees[8]

- Can we simplify the verification of the domain-level specifications?
  - It took several hours to prove the properties in [7] and couple of them timing out after 12 hours
- Can we infer new input-output specifications based on the trained models?
  - Helps in validating the model with the user, requirements elicitation

[7] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, Mykel J. Kochenderfer: Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks, 2017
[8] Idan Refaeli, *Guy Katz: Minimal Multi-Layer Modifications of Deep Neural Networks, 2020*

Proof Decomposition, Specifications Inference

- **ACAS Xu** has domain-level specifications that the network needs to satisfy
  - A => B, where A represents a predicate on the input space and B is a turning advisory
  - Proof on the full network consumes a lot of time using Reluplex

- Decomposed proofs of properties of the form A => B , using "layer patterns" σ,
  - By checking A => σ and σ =>B separately w/ Reluplex;
  - Speedup of upto 75% obtained **speedup** obtained
  - Checked property that timed out with monolithic verification

- **ACAS Xu** has meaningful input variables
  - Representing network properties in terms of input variables leads to the discovery of the specifications of the domain
  - $31900 \leq range \leq 37976$, $1.684 \leq \theta \leq 2.5133$, $\psi = -2.83$, $414.3 \leq vown \leq 506.86$, $vint = 300$, has turning advisory **COC**
  - $range = 499$, $-0.314 \leq \theta \leq -3.14$, $-3.14 \leq \psi \leq 0$, $100 \leq vown \leq 571$, $0 \leq vint \leq 150$, has turning advisory **Strong Left**
  - $range = 48608$, $\theta = -3.14$, $\psi = -2.83$, vown(full range), vint (full range) has turning advisory **COC**

Future Work

- Runtime monitors
  - Monitor for abnormal behaviors (deviations from expected behavior) based on patterns for correct and incorrect behavior
- Exploring structural coverage metrics for Neural Networks
  - Patterns extracted by Prophecy have the potential to capture the behavioral / functional / feature coverage

- Talk about certifiability, EXTENSION TO OTHER TYPES OF NETWORKS

Thank You!

https://ti.arc.nasa.gov/tech/rse/research/safednn/